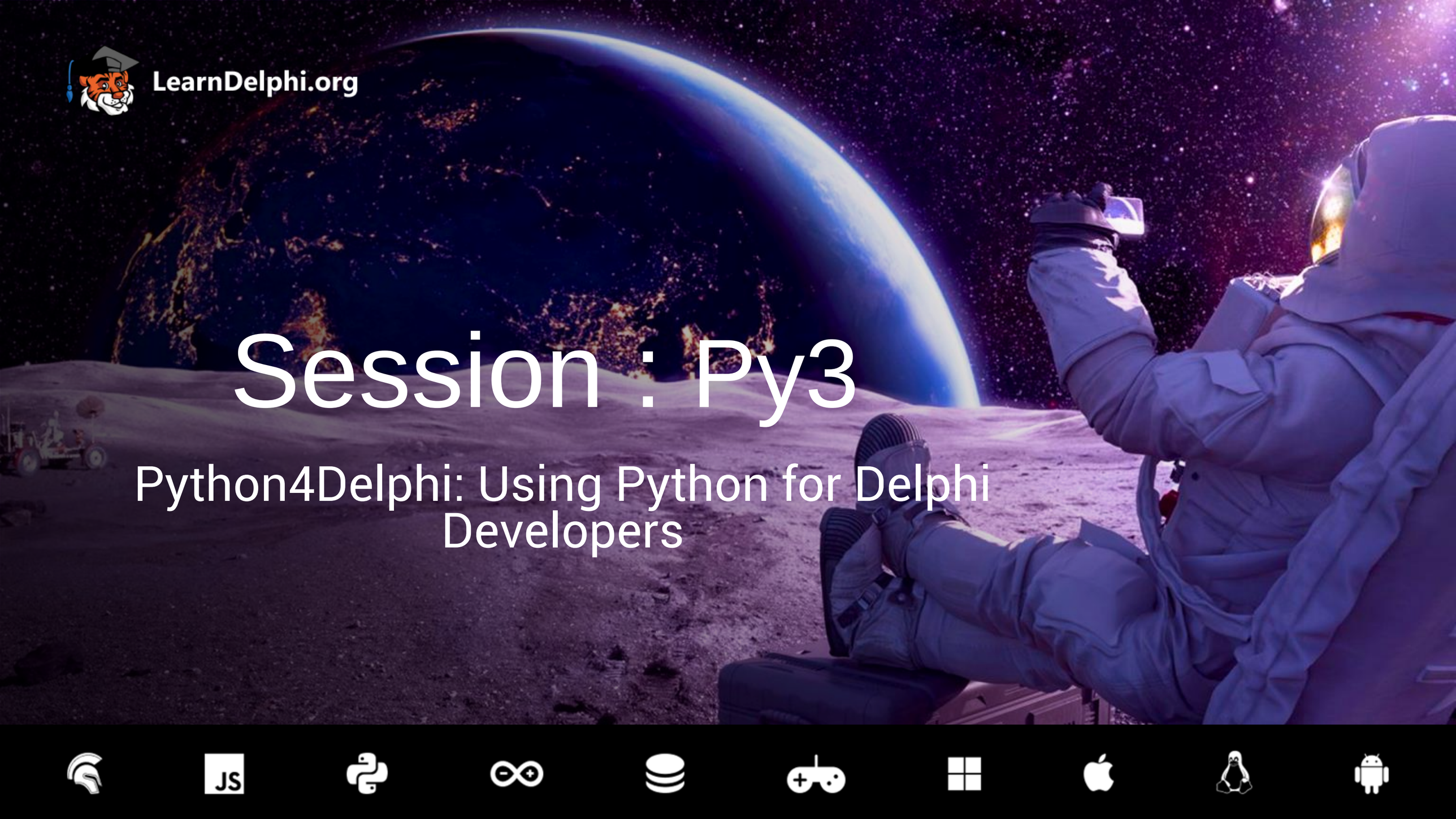
LearnDelphi.org

Coding Boot Camp

learndelphi.org/boot-camp-2022


LearnDelphi.org
Session : Py3
Python4Delphi: Using Python for Delphi Developers
JS

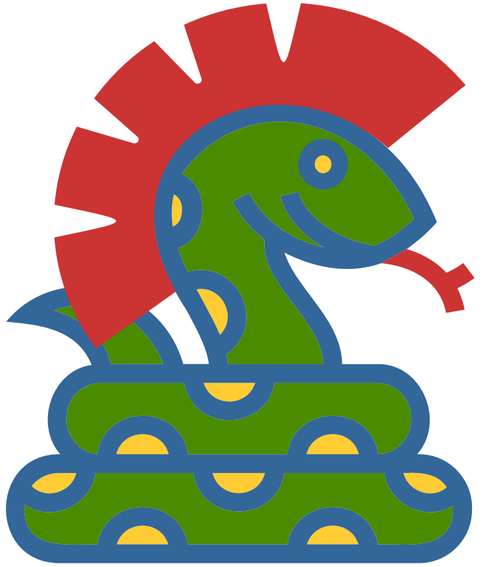
About Kiriakos Vlahos

- Professor of Management Science and Data Analytics
- Open-source enthusiast
- 35 years of programming experience (Delphi, Python)
- Developer of PyScripter
- Developer/Maintainer of many Delphi libraries, notably Python for Delphi

- Github home: https://github.com/pyscripter

Session outline

- Motivation and Synergies
- Introduction to Python
- P4D Overview
- P4D Installation
- Simple Demo
- TPythonModule
- TPyDelphiWrapper

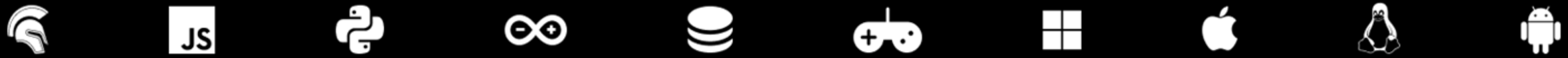


Motivation and Synergies

Python: Why should I (Delphi developer) care?

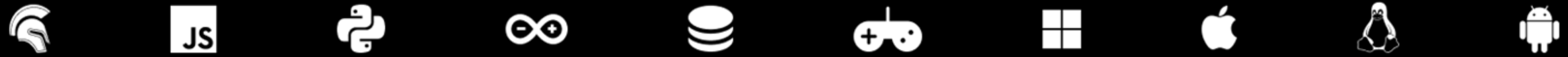
LearnDelphi.org
Coding Boot Camp 2022

- Massive popularity
- Language of choice for Data Analytics and Machine Learning/Artificial Intelligence
- Rapidly replacing Java as the core programming language in Computer Science degrees
- Huge number of packages available (250K at PyPI)
  - All the latest and greatest open-source libraries are available to Python immediately
- Perceived as productive and easy to learn
- Complementary strengths to Delphi



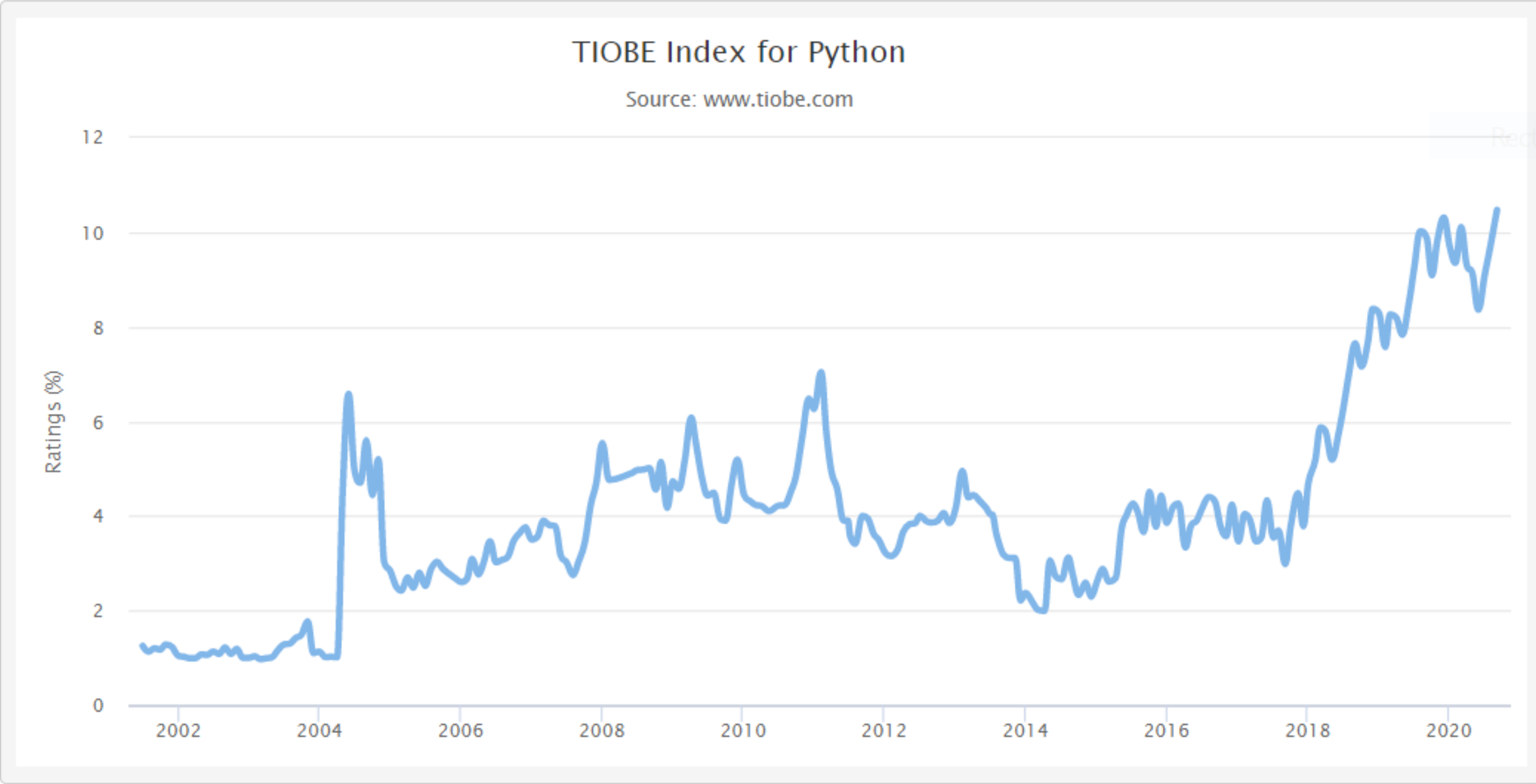
Python-Delphi: Potential Synergies

- Gain access to Python libraries from your Delphi applications
- Use Python as a scripting language for Delphi applications
- Make code developed in Delphi accessible from python scripts
- Bring together RAD and GUI Delphi development with python programming
- Combine the strengths of each language

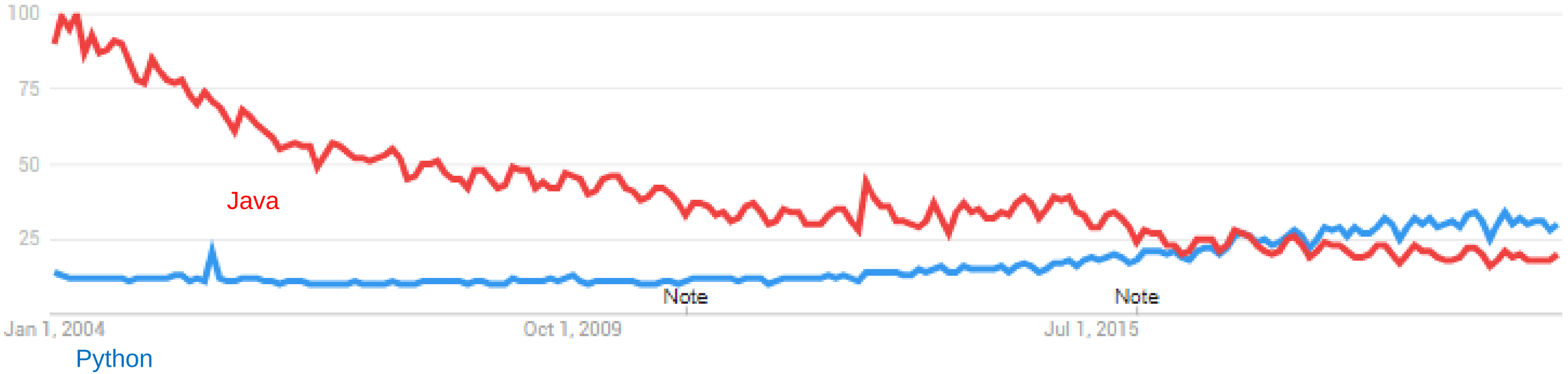


Introduction to Python

Popularity of Python



Python vs. Java - Interest over time (Google Trends)



Delphi vs Python

	Delphi/Pascal	Python
Maturity	(1995/1970!) ✓	(1989) ✓
Object orientation	✓	✓
Multi-platform	✓	✓
Verbosity	High (begin end)	Low (indentation based)
REPL	No	Yes
Typing	Strong static typing	Dynamic (duck) typing
Memory management	Manual	Reference counting
Compiled	✓	bytecode
Performance	👍	👎
Multi-threading	👍	👎
RAD	👍	👎

P4D Overview

Python for Delphi (I)

Low-level access to the python API

High-level bi-directional interaction with Python

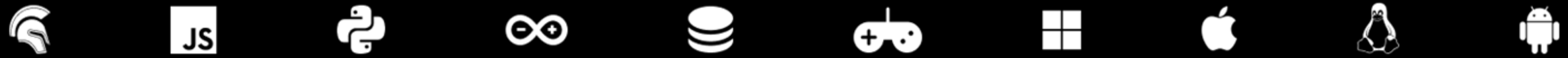
Access to Python objects using Delphi custom variants

Wrapping of Delphi objects for use in python scripts using RTTI

Creating python extension modules with Delphi classes and functions

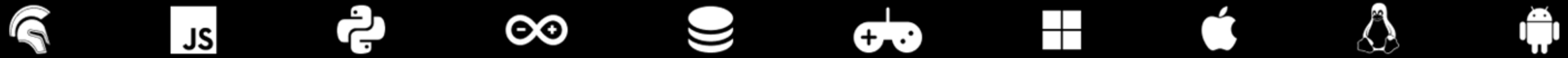
Python for Delphi (II)

- Delphi version support
  - XE2 or later
- Platform support
  - Windows 32 & 64 bits
  - Linux
  - MacOS
- Mostly non-visual components
  - Can be used in console applications
- Lazarus/FPC support



P4D Components

Component	Functionality
PythonEngine	Load and connect to Python. Access to Python API (low-level)
PythonModule	Create a Python module in Delphi and make it accessible to Python
PythonType	Create a python type (class) in Delphi
PythonInputOutput	Receive python output
PythonGUIInputOutput	Send python output to a Memo
PyDelphiWrapper	Make Delphi classes and objects accessible from Python (hi-level)
VarPyth	Hi-level access to Python objects from Delphi using custom variants (unit not a component)



P4D Installation

Getting Started – Installing Python

- Select a Python distribution
  - www.python.org (official distribution)
  - Anaconda (recommended for heavy data-analytics work)
- 32-bits vs. **64-bits**
- Download and run the installer
- Installation options (location, for all users)
- Install python packages you are planning to use (can be done later)
  - Use the python package installer (pip) from the command prompt
  - eg. > pip install numpy

Getting Started – Installing Python for Delphi

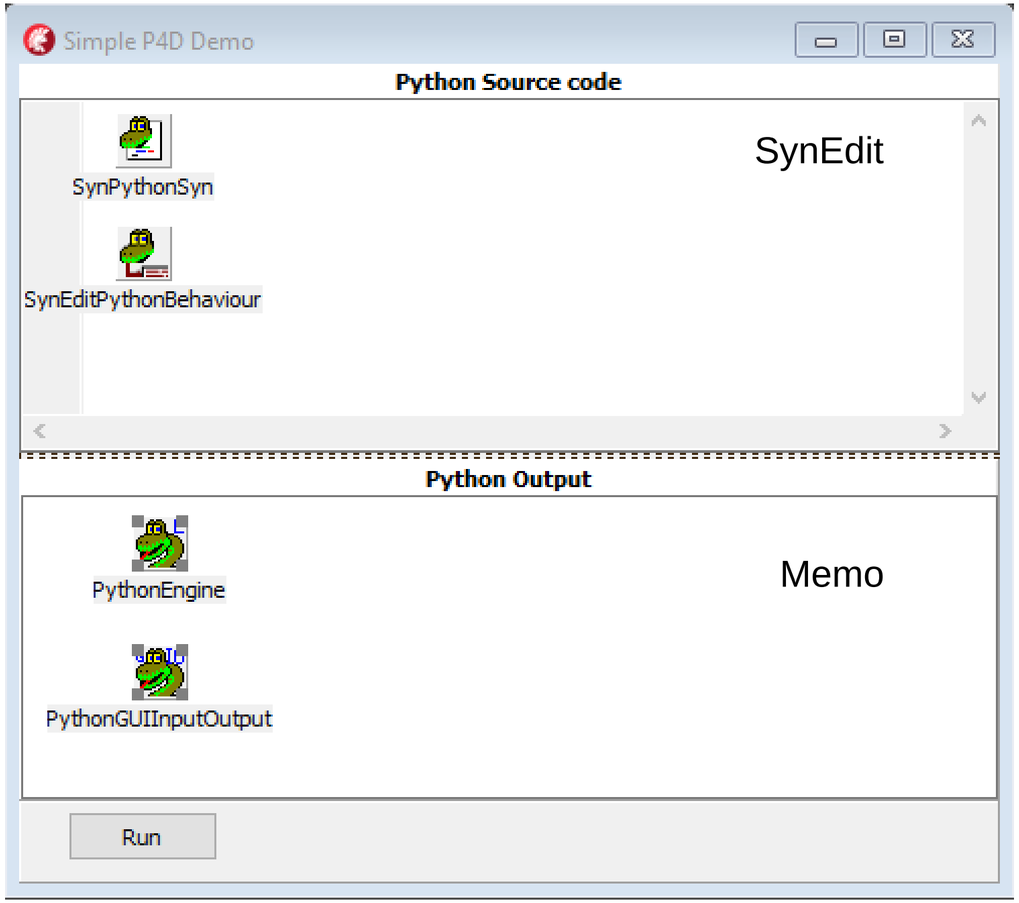
1. Use the GetIt package Manager or
2. Use the Github repo
    1. **Clone** or download and unzip the Github repository into a directory (e.g., D:\Components\P4D).
    2. Start RAD Studio.
    3. Add the source subdirectories (e.g., D:\Components\P4D\Source, D:\Components\P4D\Source\vcl and D:\Components\P4D\Source\fmx) to the IDE's library path for the targets you are planning to use.
    4. Open the "P4DComponentSuite" project group which can be found under Packages\Delphi\Delphi 10.4+ directory. Then install the dclPython, dclPythonVcl and dclPythonFmx design-time packages.

    - Instead of the steps 2.2-2.4 you can use the MultiInstaller in the Install subdirectory.

Note: The package is Design & Runtime together

Simple Demo

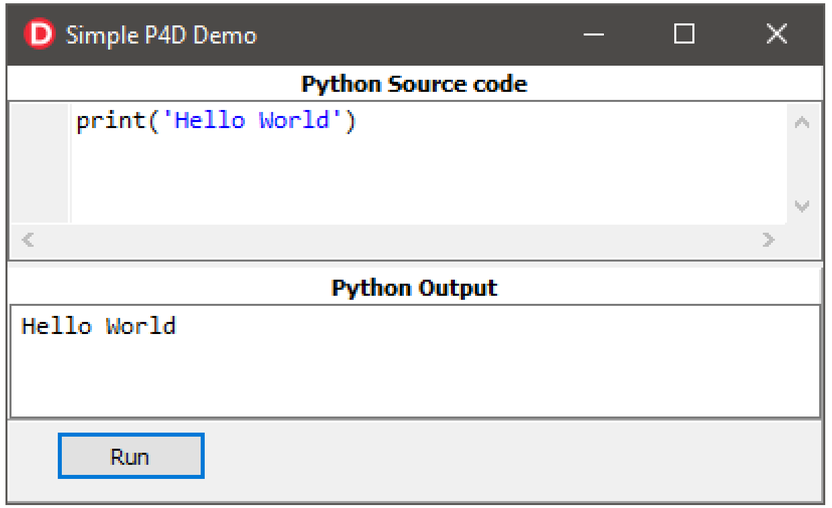
SIMPLE DEMO (I)

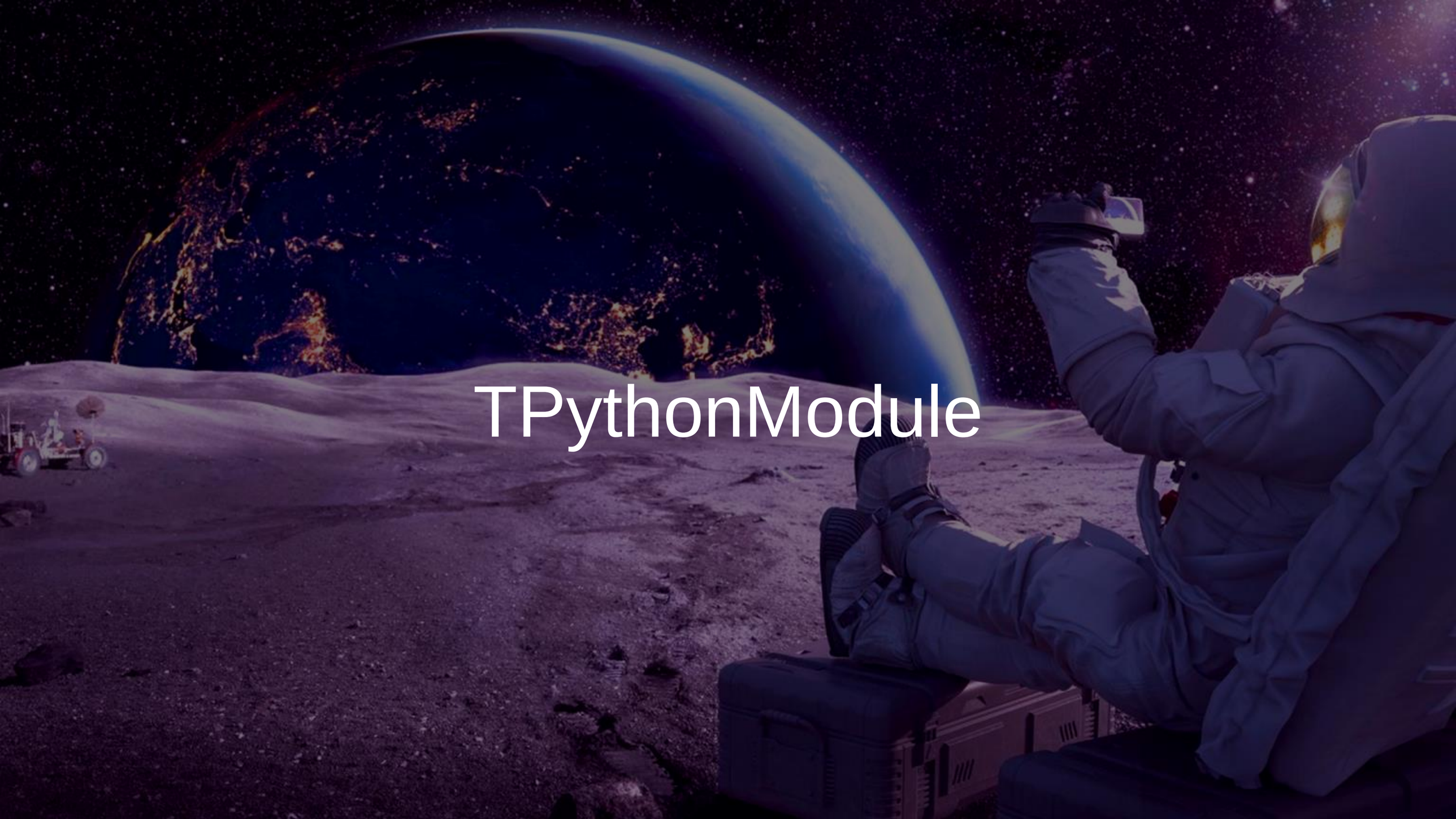


SIMPLE DEMO (II)

- All components are using default properties
  - PythonGUIInputOutput linked to PythonEngine and Memo
- Source Code:

```
procedure TForm1.btnRunClick(Sender: TObject);
begin
  GetPythonEngine.ExecString(UTF8Encode(sePythonCode.Text));
end;
```




TPythonModule

Using TPythonModule (I)

Python

```
def is_prime(n):
    """ totally naive implementation """
    if n <= 1:
        return False

    q = math.floor(math.sqrt(n))
    for i in range(2, q + 1):
        if (n % i == 0):
            return False
    return True
```

Delphi

```
function IsPrime(x: Integer): Boolean;
begin
  if (x <= 1) then Exit(False);

  var q := Floor(Sqrt(x));
  for var i := 2 to q do
    if (x mod i = 0) then
      Exit(False);
  Exit(True);
end;
```

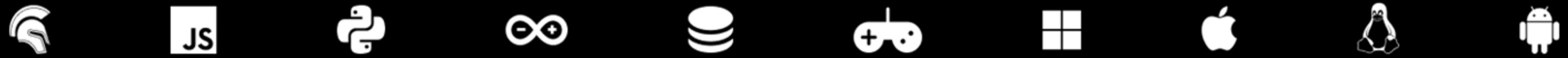
Using TPythonModule (II)

Python

```
def count_primes(max_n):
    res = 0
    for i in range(2, max_n + 1):
        if is_prime(i):
            res += 1
    return res

def test():
    max_n = 1000000
    print(f'Number of primes between 0 and {max_n} = {count_primes(max_n)}')

def main():
    print(f'Elapsed time: {Timer(stmt=test).timeit(1)} secs')

if __name__ == '__main__':
    main()
```

Output

```
Number of primes between 0 and 1000000 = 78498
Elapsed time: 3.4528134000000037 secs
```



Using TPythonModule (III)

- Add a TPythonModule to the form and link it to the PythonEngine
  - ModuleName: delphi_module
  - Implement python function delphi_is_prime by writing a Delphi event

```
procedure TForm1.PythonModuleEvents0Execute(Sender: TObject; PSelf, Args: PPyObject; var Result: PPyObject);
Var
  N: Integer;
begin
   with GetPythonEngine do
     if PyArg_ParseTuple(Args, 'i:delphi_is_prime',@N) <> 0 then
     begin
       if IsPrime(N) then
         Result := PPyObject(Py_True)
       else
         Result := PPyObject(Py_False);
       Py_INCREF(Result);
     end else
       Result := nil;
end;
```

Using TPythonModule (IV)

Python

```
from delphi_module import delphi_is_prime
def count_primes(max_n):
    res = 0
    for i in range(2, max_n + 1):
        if delphi_is_prime(i):
            res += 1
    return res
```

Output

```
Number of primes between 0 and 1000000 = 78498
Elapsed time: 0.3073742000000017 secs
```

10x + improvement!

But hold on. Delphi can do something python can't do easily: Use threads and multiple CPU cores

Using TPythonModule (V)

- Implement delphi_count_primes using TParallel.For

```
function CountPrimes(MaxN: integer): integer;
begin
  var Count := 0;
  TParallel.&For(2, MaxN, procedure(i: integer)
    begin
      if IsPrime(i) then
        AtomicIncrement(Count);
    end);
  Result := Count;
end;
```

70x + improvement!

Output

```
Number of primes between 0 and 1000000 = 78498
Elapsed time: 0.0470959000000219 secs
```

Python

```
from delphi_module import delphi_count_primes
from timeit import Timer
import math

def test():
    max_n = 1000000
    print(f'Number of primes between 0 and {max_n} = {delphi_count_primes(max_n)}')
```

TPyDelphiWrapper

Using TPyDelphiWrapper

- TPyDelphiWrapper allows you to expose Delphi objects, records and types using RTTI and cusomised wrapping of common Delphi objects.
- Add a TPyDelphiWrapper on the form and link it to a PythonModule.
- In this demo we will wrap a Delphi record containing a class function.

```
type
  TDelphiFunctions = record
    class function count_primes(MaxN: integer): integer; static;
  end;

var
  DelphiFunctions: TDelphiFunctions;
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  var Py := PyDelphiWrapper.WrapRecord(@DelphiFunctions,
    TRttiContext.Create.GetType(TypeInfo(TDelphiFunctions))
    as TRttiStructuredType);
  PythonModule.SetVar('delphi_functions', Py);
  PythonEngine.Py_DecRef(Py);
end;
```

WrapDelphi Demo31

- Shows you how you can create Delphi GUIs with Python
  - Create forms
  - Subclass Forms (and other Delphi types)
  - Add python Form event handlers
- Use customized wrapping of common RTL and VCL objects
  - Common Dialogs
  - StringLists
- Exception handling

LearnDelphi.org

Thank you!

Get the code:
github.com/Embarcadero/CodingBootCamp2022